CLAIMS

What is claimed is:

1. A method of compiling a computer program, the method comprising:
receiving a plurality of modules of source code;
generating intermediate representations corresponding to the modules;
extracting a set of data from the intermediate representations to create an inliner summary for each module; and
using the inliner summaries and a globally-sorted working-list based order in an inline analysis phase to determine which call sites in the modules are to be inlined by substituting code from a called module.

2. The method of claim 1, further comprising, after a call site is determined to be inlined:
updating a call graph of the routines (nodes) and call sites (edges); and
updating the inliner summaries throughout the call graph, as necessary.

3. The method of claim 2, further comprising, after the call graph and inliner summaries are updated,
re-calculating profitabilities associated with remaining call sites; and
re-ordering the working list using the re-calculated profitabilities.

4. The method of claim 4, wherein updating the inliner summaries comprises determining nodes and edges of the call graph that are affected by the inlining of the call site and updating those inliner summaries corresponding to the affected nodes and edges.

5. The method of claim 5, wherein the edge summaries include at least a call site execution count and a signature type.

6. The method of claim 5, wherein the node summaries include at least a code size, a routine execution count, and a call-graph height.

7. The method of claim 1, wherein the inline analysis phase is separate and distinct from an inline transformation phase.

8. An apparatus for compiling a computer program, the apparatus comprising:

a front-end portion configured to receive a plurality of modules of source code, generate intermediate representations corresponding to the modules, and extract a set of data from the intermediate representations to generate inliner summaries for the modules; and

a cross-module optimizer configured to use the inliner summaries and a globally-sorted working-list based order to analyze the call sites in an inline analysis phase so as to determine which call sites in the modules are to be inlined by substituting code from a called module.

9. The apparatus of claim 9, wherein the cross-module optimizer is further configured to update a call graph of the routines (nodes) and call sites (edges), and updates the inliner summaries, after a call site is determined to be inlined.

10. The apparatus of claim 10, wherein the cross-module optimizer is further configured to re-calculate profitabilities associated with remaining call sites, and re-order the working list using the re-calculated profitabilities, after the call graph and inliner summaries are updated.

11. The apparatus of claim 12, wherein the cross-module optimizer is further configured to update the inliner summaries by determining nodes and edges of the call graph that are affected by the inlining of the call site and update those inliner summaries corresponding to the affected nodes and edges.

12. The apparatus of claim 13, wherein the edge summaries generated by the front-end portion include at least a call site execution count and a signature type.

13. The apparatus of claim 13, wherein the node summaries generated by the front-end portion include at least a code size, a routine execution count, and a call-graph height.

14. The apparatus of claim 1, wherein the cross-module optimizer is further configured to perform the inline analysis phase separately and distinctly from an inline transformation phase.

15. A computer program product comprising a computer-usable medium having computer-readable code embodied therein, the computer program product being compiled from a plurality of modules of source code using inliner summaries and a globally-sorted working-list based order in an inline analysis phase to determine which call sites in the modules are to be inlined by substituting code from a called module.